


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide



THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

 Terms used dynamic configuration lexical analyzer

Found 19,460 of 154,226

Sort results by


[Save results to a Binder](#)
[Try an Advanced Search](#)
[Try this search in The ACM Guide](#)

Display results


[Search Tips](#)
☐ Open results in a new window

Results 1 - 20 of 200

 Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

 Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Dynamic instantiation and configuration of functionally extended, efficient lexical analysers](#)

P. Schnorf

 October 1988 **ACM SIGPLAN Notices**, Volume 23 Issue 10

 Full text available: [pdf\(721.75 KB\)](#) Additional Information: [full citation](#), [index terms](#)


2 [A graph transform model for configuration management environments](#)

Dennis Heimbigner, Steven Krane

 January 1989 **Proceedings of the third ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments**, Volume 24 , 13 Issue 2 , 5

 Full text available: [pdf\(1.32 MB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


A model for software configuration management that subsumes several existing systems is described. It is patterned after compiler models in which programs are transformed by multiple phases ending in an executable program. We model configuration management as transforming a high-level specification of a software product to be produced into a complete specification capable of being executed to construct the product. This transformational approach is used to model four existing systems and to ...

3 [Fast detection of communication patterns in distributed executions](#)

Thomas Kunz, Michiel F. H. Seuren

 November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

 Full text available: [pdf\(4.21 MB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)


Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

4 [The Pan language-based editing system](#)

Robert A. Ballance, Susan L. Graham, Michael L. Van De Vanter



January 1992 **ACM Transactions on Software Engineering and Methodology (TOSEM)**,
Volume 1 Issue 1

Full text available:  [pdf\(2.43 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)



Powerful editing systems for developing complex software documents are difficult to engineer. Besides requiring efficient incremental algorithms and complex data structures, such editors must accommodate flexible editing styles, provide a consistent, coherent, and powerful user interface, support individual variations and projectwide configurations, maintain a sharable database of information concerning the documents being edited, and integrate smoothly with the other tools in the environment ...

Keywords: Ladle, Pan, coherent user interfaces, colander, contextual constraint, extension facilities, grammatical abstraction, interactive programming environment, logic programming, logical constraint grammar, reason maintenance, syntax-recognizing editor, tolerance for errors and anomalies

5 [Parsing with flexibility, dynamic strategies, and idioms in mind](#)

Oliviero Stock

March 1989 **Computational Linguistics**, Volume 15 Issue 1

Full text available:  [pdf\(2.00 MB\)](#)  [Publisher Site](#)



Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

One desirable aspect of a syntactic parser is being meaningful (i.e., contributing to incremental interpretation) during the process of parsing and not only at the end of it. This becomes even more important when dealing with flexible word order languages, where the number of alternatives in parsing may grow dangerously. One such parser is WEDNESDAY 2. It is a lexicon-based parser, relying on the chart mechanism combined with a particular kind of unification, guided by the so-called Principle of ...

6 [A hardware/software prototyping environment for dynamically reconfigurable embedded systems](#)

Josef Fleischmann, Klaus Buchenrieder, Rainer Kress

March 1998 **Proceedings of the 6th international workshop on Hardware/software codesign**


Full text available:  [pdf\(42.66 KB\)](#)  [Publisher Site](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

7 ["Maximal-munch" tokenization in linear time](#)

Thomas Reps

March 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,
Volume 20 Issue 2

Full text available:  [pdf\(152.17 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The lexical-analysis (or scanning) phase of a compiler attempts to partition an input string into a sequence of tokens. The convention in most languages is that the input is scanned left to right, and each token identified is a "maximal munch" of the remaining input—the longest prefix of the remaining input that is a token of the language. Although most of the standard compiler textbooks present a way to perform maximal-munch tokenization, the algorithm th ...

Keywords: backtracking, dynamic programming, memoization, tabulation, tokenization

8 A full-text retrieval system with a dynamic abstract generation function

Seiji Miike, Etsuo Itoh, Kenji Ono, Kazuo Sumita

August 1994 **Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval**

Full text available:  [pdf\(802.20 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)



9 A framework for call graph construction algorithms

David Grove, Craig Chambers

November 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 6

Full text available:  [pdf\(1.36 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



A large number of call graph construction algorithms for object-oriented and functional languages have been proposed, each embodying different tradeoffs between analysis cost and call graph precision. In this article we present a unifying framework for understanding call graph construction algorithms and an empirical comparison of a representative set of algorithms. We first present a general parameterized algorithm that encompasses many well-known and novel call graph construction algorithms. W ...

Keywords: Call graph construction, control flow analysis, interprocedural analysis

10 Dynamic metrics for java

Bruno Dufour, Karel Driesen, Laurie Hendren, Clark Verbrugge

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 38 Issue 11

Full text available:  [pdf\(222.67 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



In order to perform meaningful experiments in optimizing compilation and run-time system design, researchers usually rely on a suite of benchmark programs of interest to the optimization technique under consideration. Programs are described as *numeric*, *memory-intensive*, *concurrent*, or *object-oriented*, based on a qualitative appraisal, in some cases with little justification. We believe it is beneficial to quantify the behaviour of programs with a concise and precisely ...

Keywords: Java, dynamic metrics, execution traces, optimization, profiling, program analysis, software metrics

11 Relational data bases in the design of program construction systems

S. Ceri, S. Crespi-Reghizzi

July 1983 **ACM SIGSOFT Software Engineering Notes**, Volume 8 Issue 3

Full text available:  [pdf\(968.04 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#)



A methodology for designing and implementing program construction systems using relational data bases is presented. Relations are the only data structures used by them and in the definition of interfaces between subsystems. A special-purpose relational database manager (RDB) has been designed for this application. Two large projects using this approach are described. ART (Ada-Relational-Translator) is an experimental compiler-interpreter for ADA in which all subsystems, including parser, semantic ...

Keywords: compiler design, complex software design, interface design, prototyping,


relational databases

12 The Pan language-based editing system for integrated development



Robert A. Ballance, Susan L. Graham, Michael L. Van De Vanter

October 1990 **ACM SIGSOFT Software Engineering Notes , Proceedings of the fourth ACM SIGSOFT symposium on Software development environments**, Volume 15 Issue 6

Full text available:  [pdf\(2.11 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Powerful editing systems for developing complex software documents are difficult to engineer. Besides requiring efficient incremental algorithms and complex data structures, such editors must integrate smoothly with the other tools in the environment, maintain a sharable database of information concerning the documents being edited, accommodate flexible editing styles, provide a consistent, coherent, and empowering user interface, and support individual variations and project-wide configura ...

13 Using multiple knowledge sources for word sense discrimination



Susan W. McRoy

March 1992 **Computational Linguistics**, Volume 18 Issue 1

Full text available:  [pdf\(2.02 MB\)](#)  [Publisher Site](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

This paper addresses the problem of how to identify the intended meaning of individual words in unrestricted texts, without necessarily having access to complete representations of sentences. To discriminate senses, an understander can consider a diversity of information, including syntactic tags, word frequencies, collocations, semantic context, role-related expectations, and syntactic restrictions. However, current approaches make use of only small subsets of this information. Here we will des ...

14 Conversational systems programming by incremental extension of system configuration



Rudolph A. Krutar

September 1971 **ACM SIGPLAN Notices , Proceedings of the international symposium on Extensible languages**, Volume 6 Issue 12

Full text available:  [pdf\(215.44 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


An engineer needs tools for his trade and a laboratory or toolshop in which to store them, sharpen them, and use them. A software engineer needs programming tools and a software laboratory. Input/output routines, lexical analyzers, lexicons, storage allocators, temporary files (stacks queues, sorting files), debugging aids, etc., all are useful tools in any software factory or research house. To set up a shop one needs a program library (with librarian), a linkage convention, and a set of i ...

15 Special issue: AI in engineering



D. Sriram, R. Joobhani

January 1985 **ACM SIGART Bulletin**, Issue 91

Full text available:  [pdf\(8.79 MB\)](#)

Additional Information: [full citation](#), [abstract](#)

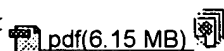
The papers in this special issue were compiled from responses to the announcement in the July 1984 issue of the SIGART newsletter and notices posted over the ARPAnet. The interest being shown in this area is reflected in the sixty papers received from over six countries. About half the papers were received over the computer network.

16 The FINITE STRING Newsletter: Abstracts of current literature

Computational Linguistics Staff

January 1987 **Computational Linguistics**, Volume 13 Issue 1-2

Full text available:



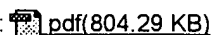
pdf(6.15 MB)

Additional Information: [full citation](#)[Publisher Site](#)**17 Technical contributions: Relational data bases in the design of program construction systems**

S. Ceri, S. Crespi-Reghizzi

November 1983 **ACM SIGPLAN Notices**, Volume 18 Issue 11

Full text available:



pdf(804.29 KB)

Additional Information: [full citation](#), [abstract](#), [references](#)

A methodology for designing and implementing program construction systems using relational data bases is presented. Relations are the only data structures used by them and in the definition of interfaces between subsystems. A special-purpose relational database manager (RDB) has been designed for this application. Two large projects using this approach are described. ART (Ada-Relational-Translator) is an experimental compiler-interpreter for ADA in which all subsystems, including parser, semantic ...

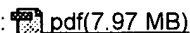
Keywords: compiler design, complex software design, interface design, prototyping, relational databases

**18 Human-computer interface development: concepts and systems for its management**

H. Rex Hartson, Deborah Hix

March 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 1

Full text available:



pdf(7.97 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

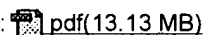
Human-computer interface management, from a computer science viewpoint, focuses on the process of developing quality human-computer interfaces, including their representation, design, implementation, execution, evaluation, and maintenance. This survey presents important concepts of interface management: dialogue independence, structural modeling, representation, interactive tools, rapid prototyping, development methodologies, and control structures. *Dialogue independence* is th ...

**19 Special issue on knowledge representation**

Ronald J. Brachman, Brian C. Smith

February 1980 **ACM SIGART Bulletin**, Issue 70

Full text available:



pdf(13.13 MB)


Additional Information: [full citation](#), [abstract](#)

In the fall of 1978 we decided to produce a special issue of the SIGART Newsletter devoted to a survey of current knowledge representation research. We felt that there were two useful functions such an issue could serve. First, we hoped to elicit a clear picture of how people working in this subdiscipline understand knowledge representation research, to illuminate the issues on which current research is focused, and to catalogue what approaches and techniques are currently being developed. Second ...

**20 Call graph construction in object-oriented languages**

David Grove, Greg DeFouw, Jeffrey Dean, Craig Chambers

October 1997 **ACM SIGPLAN Notices , Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 32 Issue 10

Full text available:  [pdf\(2.63 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Interprocedural analyses enable optimizing compilers to more precisely model the effects of non-inlined procedure calls, potentially resulting in substantial increases in application performance. Applying interprocedural analysis to programs written in object-oriented or functional languages is complicated by the difficulty of constructing an accurate program call graph. This paper presents a parameterized algorithmic framework for call graph construction in the presence of message sends and/or ...

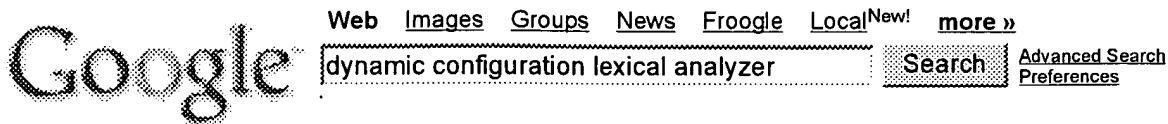
Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)







**Web**Results 1 - 10 of about **74,700** for **dynamic configuration lexical analyzer**. (0.10 seconds)Not happy with these results? [Try alternate searches](#)[RI4OS Operating systems VII: 2+2+0 VIII: 2+2+0 RI4PP Compilers VII ...](#)... **Lexical analyzer** generation. LEX generator. Context-free grammars. ...[Configuration approach to syntax analysis. Configurations for LR\(0\) and SLR\(1\) ...](#)docs.etf.bg.ac.yu/programi/ Programi_4godinaRI_engleski.html - 16k - [Cached](#) - [Similar pages](#)[ScriptBasic Users Guide: 3.4. Configuration](#)... or using **dynamic** modules you have to create a **configuration** file. ...**lexical analyzer**, syntax analyzer, code builder and external modules are not ...www.scriptbasic.com/html/texi/ug/ug_3.4.html - 13k - [Cached](#) - [Similar pages](#)[Guts: Basic Anatomy - The Perl Journal, Spring 1998](#)... with a stack of actions to be performed when leaving each **dynamic** scope. ...It's often a good idea to bypass lex and write your own **lexical analyzer**; ...www.foo.be/docs/tpj/issues/vol3_1/tpj0301-0011.html - 18k - [Cached](#) - [Similar pages](#)[\[PDF\] Title: CONVERSATIONAL SYSTEMS PROGRAMMING BY INCREMENTAL EXTENSION ...](#)

File Format: PDF/Adobe Acrobat

... **lexical analyzers**, lexicons, storage allocators, temporary files (stacks ...Here the only costs involved are suspension, resumption, and the **dynamic** ...portal.acm.org/ft_gateway.cfm?id=807991&type=pdf - [Similar pages](#)[Paper: Eclipse-2003-lee.doc :: bryant](#)... built from an existing **lexical analyzer** generator and parser generator. class... The syntax coloring for the source code in the editor is **dynamically** ...computing.breinstorm.net/ language+tg+ide+level+environment/ - 26k - [Cached](#) - [Similar pages](#)[Home Page — Guang Huang](#)... Tomasulo's algorithm(**dynamic** scheduling) simulation ... **Lexical analyzer**:Performs **lexical analysis** of the data collections ...www.csee.umbc.edu/~guangh1/r_detail.htm - 38k - [Cached](#) - [Similar pages](#)[Linux Applications](#)... DHCP (**Dynamic** Host Configuration Protocol) client daemon ... Fast, **lexical****analyzer** generator. Object-oriented scripting language. Stream editor ...www.metrowerks.com/MWW/Develop/ Embedded/Linux/applications.htm - 101k - May 8, 2005 - [Cached](#) - [Similar pages](#)[SuperMan Pages: d](#)... **Dynamic** Host Configuration Protocol Relay Agent. dhcp. dhparam. DH parametermanipulation and generation. openssl ... DFA **Lexical Analyzer** Generator ...linuxcommand.org/smp_d.php - 24k - [Cached](#) - [Similar pages](#)[Final Write-up: LoadObject](#)... I decided to take advantage of a well-known **lexical analyzer**, flex, ...I would have spent a lot more wasteful time in developing a less **dynamic** and ...www.csc.calpoly.edu/~sperry/ vecs/Writeup_loadobject.html - 10k - [Cached](#) - [Similar pages](#)[IBM: z/OS UNIX System Services ported tools](#)... shared memory management, message routing, **dynamic** (re)configuration ofdistributed services, ... flex, GNU's fast **lexical analyzer** -- similar to lex. ...www-1.ibm.com/servers/eserver/ zseries/zos/unix/bpxa1ty1.html - 42k - [Cached](#) - [Similar pages](#)

Result Page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) **[Next](#)**

Free! Google Desktop Search: Search your own computer. [Download now.](#)

Find:  emails -  files -  chats -  web history -  media -  PDF

[Search within results](#) | [Language Tools](#) | [Search Tips](#) | [Dissatisfied? Help us improve](#)

[Google Home](#) - [Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2005 Google